Sticky-Finger Manipulation With A Multi-Touch Interface

Yue Peng Toh

CMU-CS-11-124

July 2011

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Nancy S. Pollard, Chair
Jessica K. Hodgins

Submitted in partial fulfillment of the requirements for the degree of Master of Science.

Copyright © 2011 Yue Peng Toh

The author was partially supported by the Defense Advanced Research Projects Agency through the Autonomous Robotic Manipulation Software Track (ARM-S), and the National Science Foundation awards CCF-0702443 and CNS-0855171. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution or the U.S. government.

Keywords: cloth manipulation, robot teleoperation, dexterous manipulation, multi-touch, user interface, interaction, control

Abstract

Manipulation of complex simulated entities like cloth in graphical applications can be non-intuitive and awkward for the user due to the high number of degrees of freedom in their motions. In robotics, direct control of a robotic manipulator for a highly-dexterous task is similarly tedious because multiple joints have to be controlled simultaneously, in close coordination and often in varying velocities. Conventional interaction devices used in these applications, such as mice and joysticks, fail to provide a natural mapping from their relatively simple command space to the high-dimensional task space associated with the entity being controlled. In our paper, we present a user-intuitive system for the direct control of such entities using a multi-touch interface. The user manipulates simulated or real-world objects in a natural fashion by placing his or her fingers at desired points of the target objects and moving these fingers over the multi-touch surface. Any finger that is directly in contact with part of a target object becomes sticky, causing any underlying object to translate along with that finger. Our sticky finger technique allows the user to intuitively produce a range of complex manipulations such as pinching, folding, draping and tearing using common multi-touch gestures. We demonstrate our user interface in two applications, namely real-time cloth manipulation in a graphical application and the teleoperation of a human-like robot hand for common dexterous multi-fingered tasks.

Acknowledgments

Firstly, I would like to thank my advisor, Nancy Pollard for her constant guidance and encouragement over the past 2 years. I would also like to extend my thanks to Jessica Hodgins, Adrien Treuille, Gurdayal Koonjul, Garth Zeglin and the Graphics Lab for the constructive discussion involved in putting this work together.

Contents

Chapter 1

Introduction

1.1 Cloth Manipulation

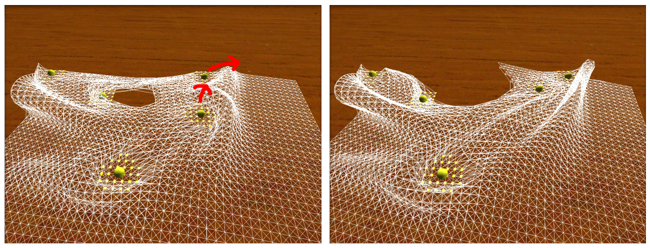

Figure 1.1: Multiple fingers are required to tear a piece of virtual cloth.

Manipulation and control of cloth in 3D scenes can often be non-intuitive and at times tedious because of its non-rigid nature. While many interactive cloth simulation systems allow for real-time manipulation of cloth by enabling the user to drag a single vertex or a subset of vertices using a 2D pointing device, this sort of dragging operation is often inadequate because only one section of the cloth can be dragged, in a single direction in 3D space at any one time.

In fact, for many useful cloth manipulations such as draping, tearing and folding, simultaneous control of different locations of a single cloth in varying directions is almost always mandatory. Precise bi-manual tearing of cloth for instance, involves pinning down sections of the cloth with a number of fingers of one hand, while moving fingers of the working hand in varying velocities on the opposite side of the desired split plane to pull the cloth apart (Figure 1.1). Likewise, draping and precise adjustment of cloth on scene models often relies on quick shifting (rapid pinning and releasing on the micro-level) of multiple fingers on the cloth surface to achieve the desired cloth configuration with respect to the model surface. Providing such global control of cloth is cumbersome and often non-intuitive with standard 2D pointing devices.

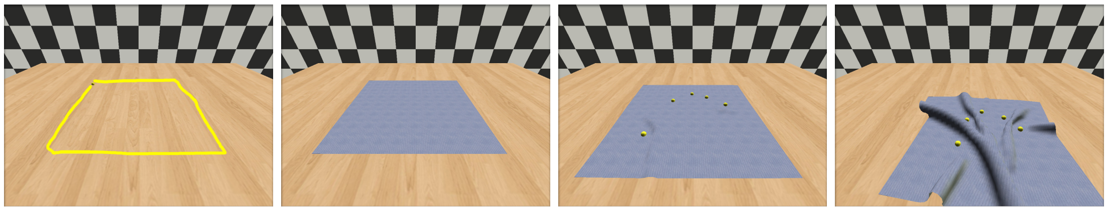
Figure 1.2: A multi-touch interface for cloth manipulation: The user first creates a cloth by drawing its shape on the scene. The user then manipulates the cloth using multiple sticky fingers.

In our paper, we address this limitation by providing an intuitive user interface for fast cloth manipulation using a multi-touch input device. The system allows the user to directly control dynamic cloth interactions by means of two main techniques: *sticky fingers* and *pinch-lifting*. For the underlying cloth simulation, we adopt a position-based approach with the use of the *Verlet integration* scheme coupled with *iterative constraint satisfaction*. The combination of sticky fingers and a position / constraint oriented integration scheme is crucial because it not only yields real-time and stable results for our system, but also empowers us with direct manipulability over particle positions, which is key to making *sticky fingers* work.

Our first technique, *sticky fingers*, allows the user to simultaneously control different regions of the cloth with any number of fingers simply by putting their fingers on the touch-screen and moving them around. The nearest underlying cloth particles in contact with a finger are positionally attached to that finger and move dynamically with it. When a particular finger is lifted from the multi-touch device, the cloth region that was previously pinned by that finger is released (Figure 1.2).

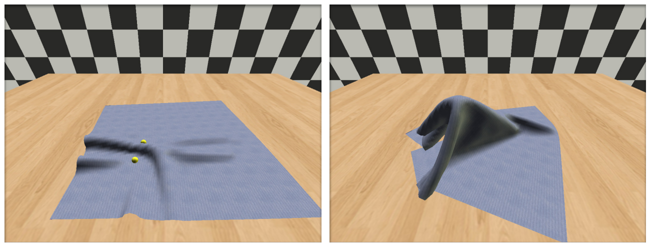
Figure 1.3: Pinch-lifting: The system interprets a pinch event as a cue to lift the cloth.

Our second technique, *pinch-lifting*, on the other hand, allows the user to intuitively lift a cloth region in a direction perpendicular to the initial plane of the cloth simply by pinching that region.

The system automatically detects a pinch event whenever the user moves two or more fingers in close proximity to one other. It then maps subsequent finger translations along the y-direction of the multi-touch device to upward movement in the corresponding scene, resulting in vertical movement of the pinched set of fingers and all the cloth particles stuck to them at that instance (Figure 1.3).

1.2 Telemanipulation

Similarly, telemanipulation of highly articulated multi-fingered robot manipulators for dexterous tasks can be equally non-intuitive with a conventional control device such as a joystick or a panel of buttons. Like in virtual cloth manipulation, the unintutiveness is due to lack of a natural mapping between the relatively simple command space of the control device and the high-dimensional task space involved in multi-fingered dexterous manipulation. While bilateral force-reflecting master-and-slave systems offer operators a more natural form of teleoperation and a heightened sense of telepresence, they are often costly, non-portable and require a substantial amount of operator learning. In addition, the bulkiness of the master manipulator can adversely affect the operator's ability to conduct the required task in a dexterous and uninhibited fashion.

In the second part of this report, we demonstrate an inexpensive, intuitive and light-weight interface for multi-fingered robot telemanipulation using a multi-touch surface. With the advent of multi-touch technology in recent years, these surfaces have become increasingly affordable, portable and accessible, which makes them cost-effective potential replacements for traditional teleoperation interfaces. Perhaps the most important benefit that a multi-touch interface offers is its directness of interaction in the form of multiple finger touches. In many existing multi-touch or tabletop applications, users are able to intuitively manipulate on-screen virtual objects by means of fingertip motions such as pinching, dragging and rotation, gestures that human use naturally for real-life dexterous tasks. In particular, direct control works by allowing the user's fingertips to seemingly come into contact and "stick" on various points on the virtual object surface, as though the virtual object is lying on the multi-touch surface. When the user moves a finger, he or she also controls the motion of the object point that was "stuck" to that moving finger. This directness empowers the user with an intuitive and physically familiar sense of interaction that cannot be reproduced with a set of buttons or a pointing device.

In our system, we leverage the concept of direct *sticky-fingers* for the telemanipulation of real objects using a multi-fingered human-like robot hand, the Shadow Hand. When the operator moves any number of fingers over the multi-touch surface, our system causes the corresponding set of robot fingertips to move in a similar fashion within a localized region on the workspace plane. By controlling the motions of the robot fingertips directly, the operator is able to intuitively reproduce multi-fingered planar manipulations such as sliding, pinching and twisting. In addition to being natural, this form of direct interaction also has the advantage of being non-obtrusive; the operator can telemanipulate with his or her bare fingertips without having to control an additional master device such as a haptic dataglove.

Our key technique of mapping multi-touch information involves treating registered finger touches as target end-effector positions for the whole robot kinematic skeleton (with branching multiple finger end-effectors) to achieve. In each time step, we used the Jacobian pseudoinverse Method to solve for the inverse kinematics of the arm and hand joints. Because the pseudoinverse method returns one out of many possible solutions, we enforced secondary goals such as orientation constraints to encourage more natural-looking robot hand poses.

We describe the details of our multi-touch interface for virtual cloth manipulation and telemanipulation in Chapter 2 and Chapter 3 respectively.

Chapter 2

Sticky-Finger Cloth Manipulation

2.1 Related Work

Cloth simulation has been a subject of unabated interest within the Computer Graphics community in recent decades. In particular, a great deal of effort has been focused on physically based approaches to cloth modeling. As a small sample of influential research in this area, Terzopoulos and his colleagues [35] introduced cloth simulation techniques to the graphics community. Breen et al. [7] performed early research on particle models with focus on physical parameters. Baraff and Witkin [6] used an implicit integration method to stably simulate realistic-looking cloth with large time steps. Choi and Ko [10] used a semi-implicit integration method to produce stable yet responsive cloth by handling post-buckling problems in cloth. Bridson and his colleagues [8] provided an efficient approach that robustly handled friction and contact responses. Their model also took into account cloth thickness.

A number of systems attempt to simulate cloth using position-based approaches instead of working purely with forces. Jakobsen [19] used Verlet Integration to gain direct manipulability on particle positions and resolve distance constraints by iterative satisfaction. Provot [29] used constraint projection as a post-correction step to fix over-elongated springs. Müller [27] adopted a complete position-based approach that handles general constraints. Meyer et al. [25] used an approximation of an implicit integration scheme coupled with position-based correction to attain stability and real-time control of virtual cloth. Many of these position-based approaches could work well with our *sticky fingers* system. We follow Jakobsen's approach [19] due to its simplicity in development and initial implementation.

Many commercial 3D applications such as garment-design programs and animation software require real-time manipulation of virtual cloth. Wojtan et al. [38] proposed a system that allows

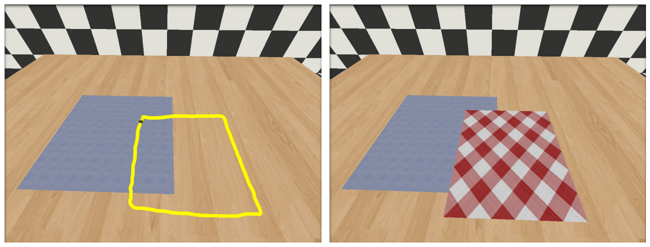
Figure 2.1: Cloth Creation: Newer cloths are automatically created on top of older ones in the event of overlaps.

keyframe control of cloth. However his system is not real-time. Igarashi and Hughes [17] created an interactive system that allows the user to wrap cloth around a virtual character and adjust cloth vertices by dragging cloth vertices along the surface of the underlying geometry. Meng, Mok and Jin [24] produced a virtual try-on system that allows pre-positioning and draping of clothes on virtual bodies by means four basic manipulations: fix, drag, move and pin. While the last two systems mentioned are real-time, each operation can only be carried out one at a time, via a solitary mouse cursor. Haptics have also been explored for manipulation of three-dimensional geometries. Notably, Dachille et al. [18] proposed a haptic system that allows manipulation of physics-based B-spline surfaces with haptic feedback.

In contrast with previous systems, this paper develops techniques for real-time multi-touch control of cloth, allowing multi-point motions such as tearing, pinch lifting, and controlled bunching, rotation, and smoothing.

2.2 User Interface Features

2.2.1 Cloth Creation

The system directly allows the user to initialize cloths of different shapes and sizes onto the scene in *cloth creation mode*. In this mode, the user simply draws cloth pieces on the multi-touch device, one after another using simple swipe gestures. Any valid cloth shape drawn is automatically converted into an appropriate cloth mesh at the scene location where the cloth is drawn. Gestures that do not resemble any supported shape such as incomplete unclosed shapes are otherwise rejected. In the event that part of a new piece of cloth is drawn over an existing cloth, the system automatically recognizes that part of the cloth to be physically on top of the corresponding area of the underlying cloth (Figure 2.1).

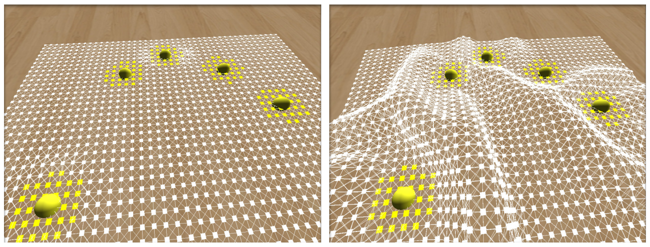

Figure 2.2: Sticky Fingers: Underlying cloth particles within a radius from the center of an active finger are stuck to that finger and move with it.

2.2.2 Cloth Manipulation

Sticky Fingers

Once a piece of cloth has been created, the user can manipulate the cloth using *sticky fingers* (Figure 2.2). The user does so first by placing any number of fingers on the multi-touch screen. Each active finger that is in contact with a part of a virtual cloth becomes sticky; All underlying cloth vertices/particles within the finger contact area are automatically stuck to that finger and are dragged along with the finger whenever the finger moves. When the finger is removed from the touch surface, the same set of particles are freed from the particular finger. By default, the fingers are positioned on a default floor plane (y = 0) in the scene and slide over the floor plane when moved. Since cloth pieces are also drawn on the floor by default, each finger that is in contact with a cloth effectively pins the cloth onto the underlying floor surface and drags that part of the cloth along the floor surface when moved.

Our dragging mechanism markedly differs from the typical vertex dragging operation in many real-time cloth simulation systems in that various sections of the cloth can be dragged and released simultaneously in varying velocities. The typical single vertex dragging operation allows the user to select a vertex or a set of vertices using a single 2D input cursor for uni-directional displacement. Dragging a single vertex tends to induce large initial deformations near the vertex and it may take a substantial amount of time for the change to propagate to the remaining vertices. Therefore, while this technique is often useful for minute local adjustments of the cloth, it is ill-suited for global manipulations such as twisting a piece of cloth around a model surface during draping.

Moreover, many manipulations such as cloth adjustment on models also require rapid shifting of fingers to pin down varying areas of the cloth while applying dragging changes to other portions of the cloth (Figure 2.3). Such precise adjustments are difficult to control with a solitary 2D input cursor. Technically, this process can be replicated by rapidly inserting and removing pins to constrain and free specified locations of the cloth while making adjustments to the rest of the cloth.

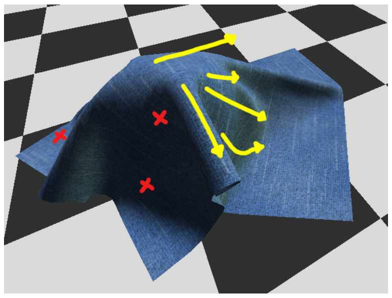

Figure 2.3: In many manipulations such as precise draping, multiple fingers play both the roles of pins and movers. Yellow arrows depict directions of dragging while red crosses represent regions where cloth should not move as the remaining free parts are being manipulated, at a particular instance. The position of these arrows and crosses change rapidly over time as the user guides the cloth into his/her desired configuration by rapidly shifting his/her fingers. Clearly single-vertex dragging is inadequate in this case.

However, such a technique is obviously non-intuitive and extremely tedious to use. Some systems actually allow simultaneous translation of multiple vertices, but this process collectively propagates the vertices in a single direction, which is clearly inadequate when we require different portions of cloth to move in varying velocities at different instances. In fact, many basic cloth manipulations such as pinching and flattening require simultaneous and multi-directional control.

Our multi-touch sticky fingers technique allows the user to independently control different parts of the cloth at different velocities. This technique also allows the user to quickly shift his/her fingers rapidly to precisely determine which parts of the cloth to pin, and which parts of the cloth should be manipulated or simply left alone to be influenced by natural forces such as gravity and drag. Unlike most conventional interactive systems, our system does not need to explicitly specify discrete basic cloth operations, such as dragging, rotation, pinning and global translation. The process of using sticky fingers implicitly supports all these features in a manner that feels intuitive and natural to the user. In addition, it automatically supports bi-manual cloth operations such as dual-pinch folding and tearing (Figure 1.1).

Sticky Finger Size

The size of each sticky finger directly determines the number of underlying particles that can be stuck to the finger. Currently, the system does not take into account the exact shape and area of finger contact on the multi-touch screen and uses a simple 2D circle to represent its area of influence on the cloth. The user may however find it useful sometimes to resize the area of the finger via an option in the user interface even though we find that the default circular area is often adequate for

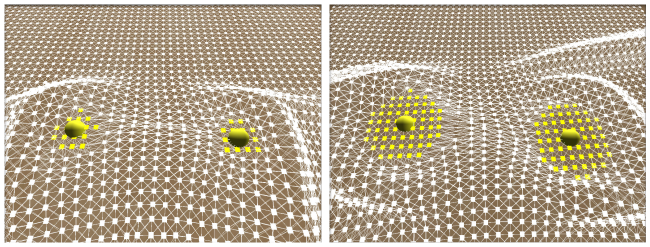

Figure 2.4: Adjustable Finger Size: On the left, small needle-like fingers pin down fewer particles and constraints. On the right, bigger fingers immobilize more particles and constraints on each side of the split plane, focusing greater stretch effects in the constraints at the zone of the desired split plane during tearing.

many manipulations.

In fact, resizing the circular area has an effect of determining how much cloth area each finger is pinning down and any one time. A small finger radius pins down fewer particles and vice versa. Planar tearing manipulation for instance, requires some number of immobilizing fingers to be holding down portions of the cloth while moving the other fingers away from the desired split plane to achieve a tearing effect. Logically, the larger the area of individual immobilizing fingers, the larger the collective area of the cloth that is immobilized. This tends to make it easier to tear a particular cloth because the user is pinning down many particles and constraints while letting the desired constraints nearer the opposite side of the split plane to elongate as far as they can until they snap instead of letting many constraints elongate bit by bit throughout the cloth (Figure 2.4).

Cloth Selection

In the event that a sticky finger comes into contact with two overlapping pieces of cloth, the finger automatically pins downs and sticks both underlying areas of the cloths to itself. This behavior is very useful if the user needs to manipulate both layers of the cloth in unison, such as tearing stacked cloths together, or simply moving a piece of cloth that has been folded to become two layers. It could represent a common situation where the friction between cloth layers is high relative to that with the underlying surface. However, the user can choose to control only the first layer of cloth that the finger encounters by enabling a special mode known as the stick to top only mode (bottom left of Figure 2.5). This mode is particularly useful for picking out and separating overlapping cloths.

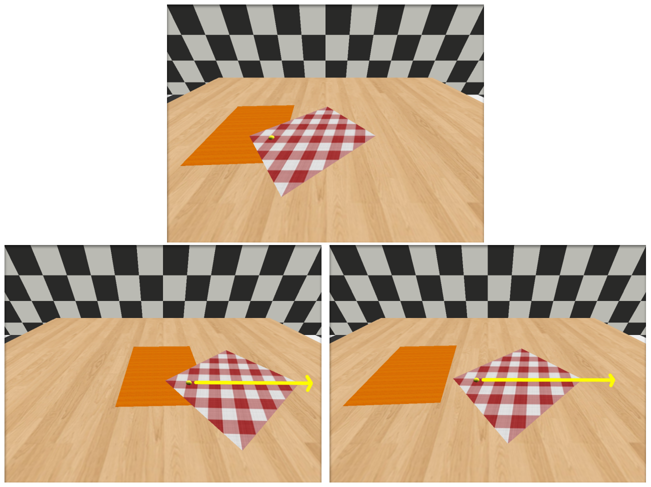

Figure 2.5: Cloth Selection: Top: A finger contacts a region with two layers of cloth. Bottom Left: When the finger moves, the default response is to move both layers together in unison. Bottom Right: However, it is sometimes useful to move only the top layer that is directly in contact with the finger.

Lifting With Sticky Fingers

Often, the user may find the need to lift sections of the cloth vertically, especially in important manipulations such as folding and draping. The user can achieve that in two ways: directly by sticky fingers or by pinch-lifting. To lift a piece of cloth by sticky fingers, the user invokes a hot-key that remaps upward motion on the physical touch pad to movement of each sticky finger up along the y-direction in the scene space. Individual fingers can then lift the cloth without having to explicitly pinch the cloth by having the particles stuck underneath them move up with them. While this is a less physically realistic maneuver, it is particular useful in situations where the user finds himself/herself running out of fingers. One example of such a situation is when the user wants to pinch all four corners of a cloth and lift the cloth. This is not possible with just two hands, so another way of achieving this operation is to use four sticky fingers, one at each corner of the cloth, to execute a upward lifting operation (Figure 2.6). The second method of lifting is realized by pinch-lifting, which is described in the next section (Section 2.2.3).

2.2.3 Pinch-lifting

While the first method of lifting a piece of cloth via sticky fingers as described in Section 2.2.2 can be convenient, it tends to be less intuitive and also less physically natural. The system hence

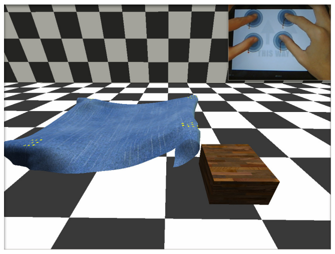

Figure 2.6: Lifting With Sticky Fingers: Each desired region of the cloth can be lifted with one finger. Compare this to pinch-lifting in Figure 1.3. The latter would require at least two fingers at each lifting point.

provides the user a more intuitive way of cloth lifting by interpreting a pinch gesture made at a portion of the cloth as an indication that the user is about to lift that bit of the cloth. When the user moves two or more fingers close enough towards each other at a point on the cloth, the system automatically switches the mode of control to vertical movement in the scene (in similar fashion to the process of invoking the hot-key for *sticky finger* lifting). Subsequently the user can slide the pinched set of fingers upwards on the touch-pad, and this movement is now interpreted as moving the cloth upwards in the scene instead of inwards in the direction parallel to the floor plane. Indeed, most of our testers find this particular mode of lifting more natural than lifting with *sticky fingers*, especially if the task is as simple as pinching two ends of the cloth and waving it about. However for more complex tasks such as precise folding and draping, most testers find it easier and more efficient to simply use *sticky fingers* for lifting.

2.2.4 Pinning

Selected cloth sections can be explicitly pinned by using the pinning sub-tool offered in *cloth manipulation mode*. The user does so by tapping on the appropriate area to push a pin through the cloth at the required location. To remove the pin, the user invokes a ”delete pin” hotkey in pinning mode and taps on the appropriate pin to remove it.

2.2.5 Adjusting the Capture Region

The user can dynamically control the size and location of the capture region, a 2D rectangular zone in scene space that represents the area of the physical multi-touch surface mapped directly to scene space (Figure 2.7). The location of this capture region determines the exact scene area where any active finger can exert its influence. This applies to both *cloth creation* and *cloth manipulation*.

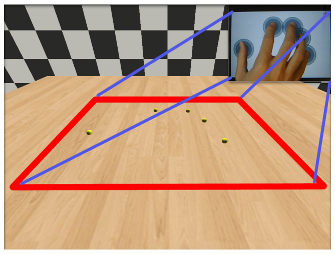

Figure 2.7: The capture region: a user-adjustable rectangular zone of control in the scene that corresponds to the area of the physical touch-pad.

The user manipulates a set of directional hot keys to dynamically resize or re-position the capture region. The default 2D capture region in the scene lies on the x-y plane and is centered about the origin (Figure 2.7). The ratio of the width to the height of the capture region is clamped by default to the aspect ratio of the physical multi-touch surface area. This is done to ensure an intuitive and more natural mapping of the physical finger movement on the multi-touch surface to that within the 3D scene. The user can over-ride this limitation by manually re-specifying the scaling factor. Re-sizing the capture region can be especially useful if the user requires very fine-tuned control over a small portion of the cloth, such as a precise pinch. In this case, the capture zone can be quickly shrunk and re-positioned at the desired area of influence on the cloth. The capture region is now represented by a larger physical multi-touch surface, allowing any finger movement on the surface to correspondingly map to a relatively finer adjustment on the virtual cloth in the scene.

2.3 Cloth Simulation Model

It is imperative that we use a fast and stable integration scheme since our system seeks to be interactive in nature. We first model cloth as a simple grid of particles connected to one another by stick-like constraints (Figure 2.8). For the particle simulation, we choose to use the velocity-less *Verlet integration* scheme with *iterative constraint satisfaction* [19] due to the speed and relative stability that this approach provides. In particular, this method works directly on positions, which is key to making our technique of *sticky fingers* work. In this section, we give a brief technical overview of the *Verlet* scheme used in the context of our system. For more in-depth background about basic *Verlet integration* with corrective constraint projection, we would like to refer the reader to Jakobsen's work [19].

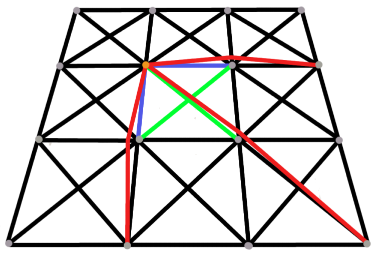

Figure 2.8: The basic cloth model with stretch (blue), bend (red) and shear (green) ”stick-like” constraints. The bend constraints are slightly warped to show the underlying constraints clearly.

2.3.1 Cloth Model

We represent cloth as a simple regular grid of particles inter-connected by various stick-like constraints (Figure 2.8). Particles immediately adjacent to each other are connected by a stretch constraint. Particles that share the same shortest diagonal are connected by a shear constraint. Finally, particles that are two neighbors away from each other in the cloth grid are connected by a bend constraint. While traditional mass-spring models often use spring-damper forces to represent constraints between particles, such an approach tends to produce ”super-elastic” cloth [29], which often looks un-realistic. Our position-based approach does not use spring forces to maintain distance constraints. Instead, we avoid the problem of over-elongation by attempting to correct the positions of particles that have moved out too far beyond their allowed constraints. We achieve this by a process known as iterative constraint satisfaction (Section 4.3). Our approach allows us to generate rather stiff ”stick-like” constraints between cloth particles, which results in more realistic-looking and less stretchy cloth.

2.3.2 Particle Integration

In most particle integration schemes, a basic particle typically maintains three variables: position $\boldsymbol{x}$, velocity $\boldsymbol{v}$ and acceleration $\mathbf{a}$. In Verlet Integration however, the current velocity is approximated by the difference between the current position of the particle, $\boldsymbol{x}_{cur}$ and its position in the previous time-step, $\boldsymbol{x}_{old}$. Therefore, velocities are now implicitly updated via direct manipulation of positions and need not be explicitly stored. Instead it becomes necessary to always maintain and update $\boldsymbol{x}_{old}$ at each time-step. The main Verlet update rule is shown in equation (1). In equation (2), the old position, $\boldsymbol{x}_{old}$ is updated with the value of the current particle position $\boldsymbol{x}_{cur}$, so that it can be used in the next time step.

$$\mathbf{x}_{new} = 2\mathbf{x}_{cur} - \mathbf{x}_{old} + \mathbf{a}\Delta t^2 \tag{2.1}$$

$$\mathbf{x}_{old} = \mathbf{x}_{cur} \tag{2.2}$$

2.3.3 Iterative Constraint Satisfaction

The final requirement for our interactive system is to be relatively stable. This is achieved via iterative constraint satisfaction. After the Verlet integration step, particles attain new positions. Multiple pairs of particles may have been over-stretched beyond their desired rest distance. To keep cloth structure intact while allowing the cloth to exhibit desirable properties such as shearing and bending, we attempt to satisfy all the constraints mentioned in 4.1 by relaxation. We iterate through all the distance constraints a number of times and attempt to restore pairs of particles back to their rest states by applying suitable position corrections to each particle. For any constrained pair of particles with positions $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, and initial rest distance r between the particles, the amount of correction for each particle is computed using formulas (3) and (4) as proposed by Jakobsen. In (3), we compute the total displacement that particle $\boldsymbol{x}_1$ alone has to make to be one rest distance apart from particle $\boldsymbol{x}_2$. This correction vector is expressed as follows:$(|\mathbf{x}_2 - \mathbf{x}_1| - r)\frac{\mathbf{x}_2 - \mathbf{x}_1}{|\mathbf{x}_2 - \mathbf{x}_1|}$. We then distribute this correction vector to both particles, weighing their respective corrections by the ratio of their individual masses to their combined masses (In our case, since we consider only cloth of uniform material, the masses are equal and hence each particle gets exactly half the distance correction) and attempt to move each particle towards each other by their calculated corrections.

$$\Delta\mathbf{x}_1 = \frac{1}{2}(|\mathbf{x}_2 - \mathbf{x}_1| - r)\frac{\mathbf{x}_2 - \mathbf{x}_1}{|\mathbf{x}_2 - \mathbf{x}_1|} \tag{2.3}$$

$$\Delta\mathbf{x}_2 = -\Delta\mathbf{x}_1 \tag{2.4}$$

2.3.4 Integration with Sticky Fingers

Integrating sticky fingers with our simulation model is generally seamless but does require a little book-keeping. Specifically, during iterative constraint satisfaction, we need to ensure that any constraint that involves a cloth particle that is already stuck to a sticky finger does not try to move the particle. On the other hand, the particle on the other ”end” of the constraint should attempt to obey that constraint by moving towards the stuck particle, only if it is not stuck to a finger as well. To handle this situation, each particle is flagged whenever it is stuck to a sticky finger. During

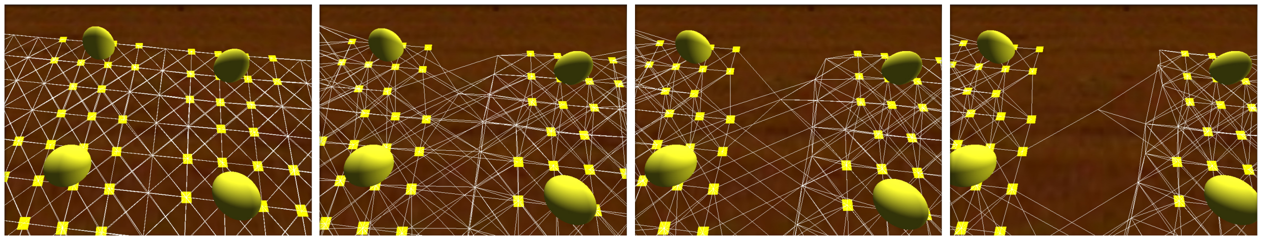

Figure 2.9: Simple mesh tearing is achieved by removing constraints that are ”over-stretched” beyond an allowed distance threshold

iterative constraint satisfaction, for each constraint that contains a flagged particle $\boldsymbol{x}_1$ and unflagged particle $\boldsymbol{x}_2$, we compute the total correction vector as in 4.3 but distribute the full correction vector entirely to $\boldsymbol{x}_2$ to ensure that the stuck particle is fixed to the finger while only the unflagged particle attempts to move towards the stuck finger. Essentially, we are setting the mass of stuck particle to infinity, since its motion is clamped to that of the finger and should never be allowed to obey any other constraint until the finger is lifted. Formally, the corrections are expressed as follows:

$$\Delta \mathbf{x}_1 = 0 \tag{2.5}$$

$$\Delta \mathbf{x}_2 = -(|\mathbf{x}_2 - \mathbf{x}_1| - r)\frac{\mathbf{x}_2 - \mathbf{x}_1}{|\mathbf{x}_2 - \mathbf{x}_1|} \tag{2.6}$$

In the event that both particles are being stuck to fingers, both correction values should be set to zero. In the case where both particles are not stuck, equations (3) and (4) from the previous section should be applied.

Explicit pinning of cloth regions by the user is handled exactly in the same manner; These particles are flagged as being ”stuck” whenever they are pinned, and unflagged when the pin is removed.

It is worth mentioning that our equations are identical in spirit to the constraint projection equations that involve inverse masses [27]. To stick particles to moving objects, Müller zero-ed the inverse mass of the particle and weighed the respective correction vectors accordingly [27]. We found it slightly easier though to flag ”stuck particles” and treat the three possible cases individually instead of explicitly using and modifying inverse masses.

2.3.5 Tearing

We allow tearing of a cloth mesh by proceeding with a simplified approach of removing constraints containing particles ”stretched” apart by a certain allowable distance threshold (Figure 2.9). While many other robust tearing methods such as the particle-splitting method [27] exist, our simplified model is easy to implement and yields satisfactory visual results for higher resolution meshes. The feel of the cloth can also be varied by the user simply by specifying the maxmimum tearing threshold for each kind of constraint. If the threshold values are generally low with respect to the initial rest distances between cloth particles, the cloth exhibits highly brittle behavior, even upon mere contact. Conversely, if the thresholds are high, the resulting cloth is very hard to tear and may have to be stretched by a large distance before the first instance of tearing occurs.

2.4 Implementation Details

The system is implemented in C++ and uses OpenGL for rendering. The system is tested on a Intel Core 2 Duo laptop at 2.0 Ghz. The exact number of particles depends on the dimensions of the cloth as drawn by the user but a cloth mesh with a typically high resolution can consist of about 1000 particles on average. Constraints are satisfied 5 times per frame. Even with 1000-particle cloth meshes, we are able to attain highly satisfactory frame rates for most of the manipulations carried out. The cloth structure is quite robust in terms of stability due to the process of iterative constraint satisfaction at each time step. If the user somehow manages to blow up the cloth or cause a large number of self-intersections, the cloth automatically recovers and unravels itself neatly when left alone for a short amount of time.

2.4.1 Multi-Touch Integration

We use the TUIO protocol [20] for multi-touch integration with our system. This protocol allows the transmission of touch events in the form of UDP packets from a suitable finger tracker application to our system. On start-up, our system listens on a UDP port for these touch messages. A basic touch message typically consists of the position, velocity and area of each individual finger that is in contact with the multi-touch surface. Our system constantly keeps track of position information of each physical finger which is given in normalized touch-screen coordinates, and calculates its corresponding virtual finger position on the capture region in the scene (Figure 2.7). When a physical finger moves, this calculation is updated accordingly based on the new position. Finally, when a physical finger is lifted from the touch surface, we simply delete the virtual finger from our system. Depending on the multi-touch hardware used, the area and shape of each finger that

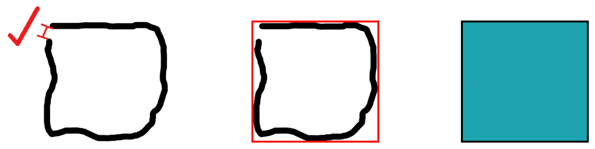

Figure 2.10: Simple Rectangle Gesture Recognition: The gesture is first tested for reasonable closure. Then an axis-aligned bounding box is constructed around the gesture. Finally the cloth is created based on the bounding box.

is in contact with the multi-touch surface may be available to the system. This information could produce more accurate physical results because the underlying cloth area of influence under the virtual finger can then be shaped according to the physical size of the actual area of contact instead of being fixed as a round finger diameter. However, our current hardware implementation does not return area information, so our system works with adjustable finger diameters instead. In fact, this setting generally works very well for all our manipulations.

Currently, we use the Apple Ipad as our multi-touch device because of its good screen resolution, reliability and portability. We also rely on a TUIO-compatible finger-tracker application for Ipad, MSA Remote [2], to track finger positions on the Ipad touch-screen and send these touch messages wirelessly to our system. Prior to choosing the Apple Ipad, we built a simple DIY multi-touch device based on diffused illumination [32] and tracked finger shadows using a camera. Using an open-source image processing library, TouchLib [28], we were able to feed the relevant finger positions into our system and use them to manipulate cloth. However, the ability to achieve satisfactory results often depend on the surrounding lighting conditions. The touch data received were sometimes noisy, which often resulted in ”ghost fingers”. We eliminated these noise issues and the hassle of constant re-calibration by switching to the capacitive touch-screen of the Ipad, which allows us to achieve consistent and robust results.

2.4.2 Simple Shape Gesture Recognition

In creation mode, the user creates a cloth by drawing the cloth shape on the multi-touch screen with a single and continuous finger stroke. Our system currently performs simple axis-aligned rectangle gesture recognition (Figure 2.10). Whenever a finger delete event occurs, we check whether the removed finger was the only active finger in the previous session. A session is defined as a time period during which the number of active fingers on the multi-touch surface does not change. If the removed finger was indeed the only active finger, the user must have executed a single and continuous swipe gesture.

Assuming that the user has attempted to draw something resembling a square or a rectangle,

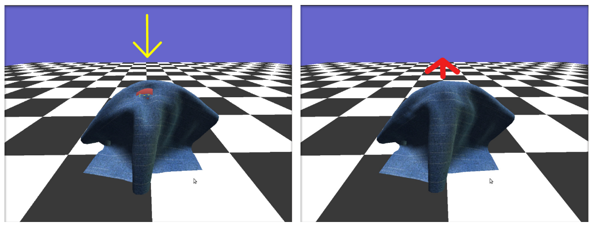

Figure 2.11: Collision Handling by Projection. The system automatically corrects any geometric violation in next time step after the user releases his finger.

we test the drawn finger path for proper closure. If the beginning and the end of the stroke are too far away from each other, we reject the gesture as an incomplete shape or an accidental stroke. Otherwise, we proceed to compute an axis-aligned bounding box from the minimum and maximum x and y values seen from the points in the finger trajectory. We then calculate the ratio of the dimensions of the resulting bounding box to the dimensions of the touch-screen, derive the scene dimensions of the cloth and initialize the cloth within the capture region on the scene. We find that this simple axis aligned bounding box approximation works surprisingly well.

2.4.3 Collision Response

To maintain our system at interactive rates, cloth self-collision is not implemented. For collision detection and response with simple scene geometries, we use a simple projection scheme that tests whether each cloth particle lies within the model. If a particle is found to be in that state, we simply push the particle onto the nearest point on the object surface. We also keep the cloth a small distance away from the surface to prevent depth-related rendering artifacts as well the possibility of edges penetrating the underlying surface. Typically though, edge penetration seldom happens because our cloth model is of a sufficiently high resolution. As for friction, we compute the tangential component of the collision response from the vector $\boldsymbol{x}_{new}$-$\boldsymbol{x}_{old}$ and the surface normal. After projection to the surface, we then scale this tangential component by a suitable frictional coefficient and attenuate $\boldsymbol{x}_{new}$ with the scaled tangential component to produce a suitable frictional response between the cloth and the surface.

Our dragging operation is unconstrained in that if the user moves his fingers too quickly into the geometry of a scene object, the cloth may also move into the object. However, our system automatically corrects this violation by projecting the cloth to the surface of the object in the following time step the moment the user releases his finger (Figure 2.11).

2.5 Results and Evaluation

Figures 2.12 and 2.13 show the basic actions of bunching, spreading, translation, and rotation performed using our system.

We evaluated our system by conducting informal user tests. We designed each test as a set of simple manipulation tasks for the user to achieve. The test begins with a ”basic functionality” task. For this task, users were requested to accomplish a series of simple planar operations, namely bunching, spreading, translation and rotation in this specific order. Generally, the average user took about 25 to 30 seconds on the first try to get a good feel of the system and complete this sequence. On subsequent attempts, most users were able to perform the required sequence under 10 seconds.

The next task involves planar tearing of a piece of cloth into two halves. We wanted to gauge how quickly and intuitive it is for the user to grasp the feel of the cloth material and also how well the user can control the tearing process. In the first few attempts, users generally have difficulties tearing a cloth into exactly two halves. They tend to produce many small scraps in the process. This is actually because the system can only utilize fingertip contact areas. In real tearing, many parts of the cloth are immobilized by large areas of the hand such as entire length of fingers, or the palm, while cloth particles near the desired split plane are unconstrained and allowed to be moved apart. Therefore, users often produce small scraps the size of their finger tips, because these are the exact areas where the cloths are not allowed to be split. When we enlarge the finger size, users tend to have higher success rates. Also, some users adapted to this limitation by interleaving rapid movement of fingers in the direction parallel to the desired split plane with perpendicular tearing away from the split plane to control the tearing process.

When we vary the cloth material to make the cloth harder to tear, users were able to ”feel” the difference in material after an initial attempt to pull it apart. We observed an interesting common strategy that emerged; Users tend to drag cloth particles away from the desired split plane then quickly shift their fingers to get a re-grip on cloth particles nearer the split plane to repeat the process, until the cloth tears. This ”clawing” strategy seems to be particularly effective because a single drag may not produce significant extensions near the split plane if the tearing thresholds are high. Multiple drags are often required to produce the overall elongation required to stretch particle distances, especially those near the split plane, beyond these thresholds.

The remaining tasks, namely draping and folding, involve explicit lifting of the cloth from the underlying plane. Users were encouraged to experiment with both pinch-lifting and lifting by sticky fingers to see which method suits them the best for each of the tasks. Most users have no problems making a simple fold or drape with no particular preference for either method. This seems to be so because the manipulation can typically be done by lifting at most two points of the cloth. When we enforced the requirement that the draping must be done precisely so that the cloth is positioned in

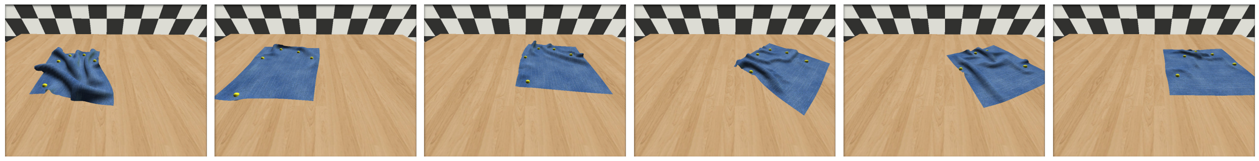

Figure 2.12: Planar manipulations. Top row: Bunching, spreading and translation. Bottom row: Three frames of rotation

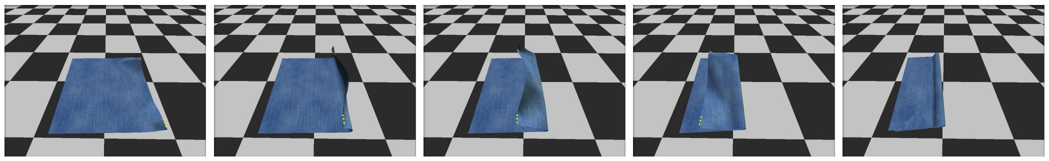

Figure 2.13: Simple Folding

a precise configuration with respect to the underlying geometry, many users tend to prefer lifting by sticky fingers because they could lift up the cloth completely with multiple fingers and position it precisely above the geometry before placing it carefully on top of the model. But a problem commonly faced with this mode of lifting is the need to trigger the toggle key for lifting. This broke the ”momentum” of the entire manipulation process, especially for users who were using sticky fingers from both hands; they find themselves putting down the cloth for a moment to trigger the toggle, before continuing. On the other hand, pinch-lifting felt more natural for most users even though they tend to do a lot more adjustments to get the cloth in the desired configuration once they manage to get it onto the model.

In general, our testers liked the novelty of our system. They also commented that cloth control felt natural and physically realistic; They were able to relate the effects of our supported operations to real cloth manipulation. However, some testers pointed out that the system would be more appealing if cloths of various shapes can be created. Other testers also felt that *pinch-lifting* is visually more believable than lifting with *sticky fingers*, even though the latter may in some cases be easier to use. In addition, a tester remarked that our interaction techniques would be very suitable for games that involve manipulation of flexible virtual objects, possibly by multiple users, since our system does not place any restriction on the number of *sticky fingers* one can place on a cloth.

Chapter 3

Sticky-Finger Telemanipulation

3.1 Related Work

Teleoperation is the control of a robot by a human operator to perform tasks in a remote environment. Being able to carry out tasks remotely can be valuable, especially when the remote environment involved is hazardous, uncertain or inaccessible to humans. As such, teleoperation systems have found a plethora of applications in space, undersea, surgical, and military operations, as well as in virtual reality [33].

More recently, significant research has been focused on developing systems for the dexterous telemanipulation of robot extremities, in particular the hand. These systems are often bilateral [15]; the operator manipulates a ”master” manipulator which structurally resembles the ”slave” manipulator to be controlled. The ”slave” hand manipulator in turn reflects haptic feedback in the form of contact forces to the ”master”, allowing the operator to gain a augmented sense of telepresence during the course of performing the dexterous task.

The bulkiness associated with many of these traditional master manipulators prompted some researchers to focus on more portable alternatives for dexterous telemanipulation, such as the dataglove. The dataglove is a glove-like device which records the degree of bending of the operator’s finger joints. Glove data can be combined with haptic feedback to control the robot hand efficiently for telemanipulation tasks [36][16]. Associated strategies for mapping between human and robot hand workspaces have been discussed in [31] and [12]. In other non-obtrusive systems, vision has also been employed to track the operator’s hand during dexterous telemanipulation [22].

Some systems have explored mapping low dimensional commands to complex manipulation trajectories, with simple input devices such as joysticks instead [34]. These approaches also emphasize transparency as well sharing control with the robot [13]; the human operator focuses on issuing high-level commands to the robot, and is isolated from the low-level complexities involved in the actual trajectory generation.

In our paper, we develop techniques for telemanipulating a robot hand dexterously and intuitively using a lightweight multi-touch input device. While multi-touch interfaces for robot control have been explored in various recent work [26][21], these systems are typically geared towards high-level teleoperation tasks such as controlling the movement of mobile robots. To the best of our knowledge, no prior system has employed a multi-touch interface for the dexterous control of a multi-fingered robot hand. On the other hand, the use of sticky fingertips for the direct manipulation of virtual objects with a multi-touch interface has been discussed extensively in [14], [30] and [37]. Our work applies the essence of sticky-finger interaction to the manipulation of real objects by proxy, via the fingertips of the teleoperated robot hand.

Many applications in computer animation and robotics employ numerical-based methods to solve various inverse kinematics problems [9]. In particular, a number of authors have used the Jacobian pseudoinverse method, along with secondary constraints to encourage a particular quality in their solution, such as collision avoidance and joint limit avoidance [23][1][5]. In our work, we introduce similar secondary constraints such as enforcing a natural palm orientation to ensure naturalness and manipulability of the mapped robot hand pose.

3.2 System Setup

Our robot system consists of the Shadow Hand mounted on the right arm of a 7-DOF Motoman SDA10 robot. The Shadow Hand has a kinematic structure that resembles that of a human right hand with 24 degrees of freedom. The finger joints are position-controlled but have some degree of compliance. The operator teleoperates the robot hand using the Apple iPad, a capacitive-based multi-touch surface. During each time frame, touch events are sent wirelessly to our main mapping program, which computes the relevant arm and hand poses for the robot, and relays the computed joint positions to the arm and hand controllers respectively (Figure 3.1). Specific details are described in Section VI-A.

3.3 User Interface Features

3.3.1 Sticky Fingers

In order to execute a sticky-finger telemanipulation, the operator begins by placing any number of fingers on the multi-touch surface. The system then assigns each active touch it sees to a certain

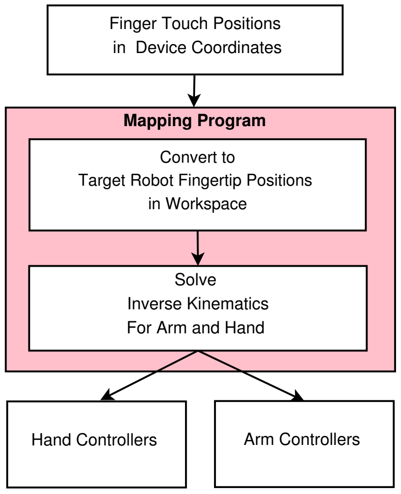
Finger Touch Positions in Device Coordinates
Mapping Program
Convert to Target Robot Fingertip Positions in Workspace
Solve Inverse Kinematics For Arm and Hand
Hand Controllers
Arm Controllers

Figure 3.1: System Overview

finger that it thinks is registering that touch. To do this in a natural and unambiguous fashion, we simply require the operator to register touches sequentially (Figure 3.2). That is, the first touch is always with the thumb, followed by the index finger and so on, till the little finger. This mapping order is also consistent with our observation of the importance of each finger with respect to performing *sticky-finger*-style manipulations. We observe for instance that a two-pinch is more commonly executed using the thumb and the index finger than perhaps the ring and little finger.

Each active finger added to the touch screen causes the same robot fingertip to move to a corresponding mapped position within a target control region in the workspace (Figure 3.3). By default, the plane of this control region is level with the workspace plane itself, but its height can be adjusted dynamically if the operator needs the robot fingertip to land on a target object surface at a different height above the workspace plane. Whenever the operator moves a finger laterally across the multi-touch surface, the corresponding robot fingertip executes the same planar trajectory as the operator's fingertip on the control region. Overall, our interaction scheme allows the operator's finger motions to be mimicked in near real-time by the robot fingers. Useful telemanipulations such as grasping, twisting and sliding can thus be reproduced by performing similar finger gestures on the multi-touch surface.

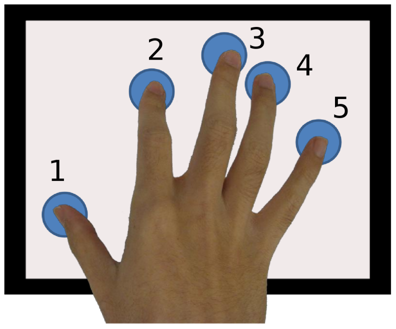

Figure 3.2: Sticky-finger Registration: To avoid ambiguity, the operator is required to register finger touches sequentially, starting from the thumb.

While we term our interaction technique as being ”sticky”, ”stickiness” is more of a convenient metaphor for the operator to have in mind while using our multi-touch interface - in particular, the idea that the operator can have direct multiple control over various control points on the object using any number of fingertips. Unlike for the manipulation of virtual objects though, the actual robot fingertips that come into contact with a real object grip or ”stick” only if there is sufficient friction generated between the fingertip and the contact point on the object surface. The resulting outcome of any such interaction hence depends significantly on the physical properties of the target object and its environment.

3.3.2 Adjusting the Control Region

By default, all sticky-finger manipulations are executed within a localized control region, a two-dimensional rectangular zone on the workspace representing the area of the physically multi-touch surface (Figure 3.3). The location of this control region determines the exact workspace area where any active finger placed by the operator can exert its influence. We currently set the dimensions of the control region to match those of the actual touch screen interface, to ensure a more direct and intuitive mapping for the operator. Here, we also made the assumption that the kinematics of the Shadow Hand and those of the operator's hand are close enough such that regardless of the operator's desired target fingertip locations, the Shadow Hand is always able to attain a corresponding

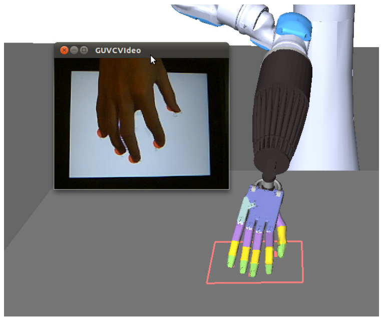

Figure 3.3: The Control Region: a user-adjustable rectangular zone of control (in light red) in the workspace that corresponds to the area of the physical touch-pad. Target fingertip positions all lie on the plane of the control region.

pose with these very same fingertip positions on the actual control region without kinematic difficulty (details in Section V).

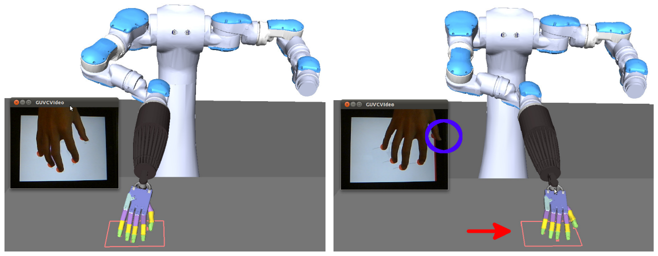

Figure 3.4: Edge Scrolling: The operator moves any fingertip, in this case the thumb (circled in blue), against a particular edge of the multi-touch surface to ”scroll” the hand in the direction beyond that edge.

Lateral Control

The operator might often want to carry out a telemanipulation task beyond the boundaries of the default control region on the workspace. An example would be to drag a piece of cloth from one end of the table to another. Adding buttons onto the touchscreen to control the location of the control region not only consumes valuable screen estate but also disrupts the flow of the teleoperation task.

For our system, we introduce a more natural way of adjusting the control region by means of *edge-scrolling*. To move the hand laterally across the workspace plane, the operator moves any finger against an edge of the multi-touch screen. Our system interprets any finger touch near one of the four boundaries as a cue that the operator wants to ”scroll” the control region in the direction beyond that particular boundary. The system then proceeds to move the control region with a fixed speed in that designated direction, which causes all active finger targets within the control region, and thus the hand to be translated in unison at the same designated speed (Figure 3.4). To prevent accidental scrolling when a touch is near an edge, the system ignores border touches that do not linger for more than a time interval of 0.5 seconds. Scrolling halts when the finger is removed from the touch surface or moved away from the associated border.

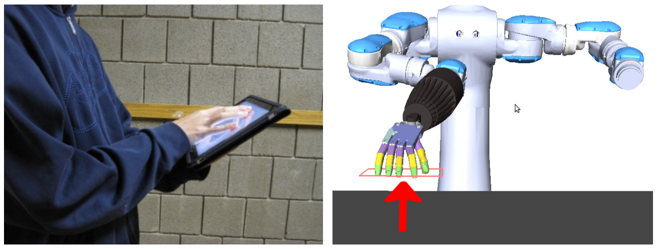

Figure 3.5: Device Tilting: The operator tilts the device and holds it beyond a predefined angle to ”scroll” the hand upwards.

Vertical Control

The default height of the control region is set to height of the given workspace. This is useful when the operator carries out sticky-finger interactions with typically thin and flat objects, which lies near the workspace plane. However when the required teleoperation task involves lifting the target object, the operator will need to be able to dynamically adjust the height of the entire hand. This is done by altering the height of the control region so that the new target fingertip positions are now lying on a different plane height than before.

The operator can control this height intuitively by altering the pitch angle of the device. Since the teleoperation is carried out on one robot hand, the operator can afford to hold the device flat on one palm, while he or she teleoperates with the other hand (Figure 3.5). The control region can be ”scrolled” upwards at a designated constant speed as long as the device is held at a pitch angle greater than the a certain designated angle, θ. The reverse motion occurs when the device is held at a pitch angle of $-\theta$. An angle value lying within the ”deadzone” range of $-\theta$ and θ results in no movement; the operator can hence halt any scrolling by restoring the pitch of the device to a more or less flat configuration. The ”deadzone” is also essential to prevent accidental scrolling. The minimum and maximum heights of the control region is clamped at the height of the workspace surface and a reasonably safe height above the table surface respectively, but the latter can be adjusted easily based on the nature of the teleoperation task.

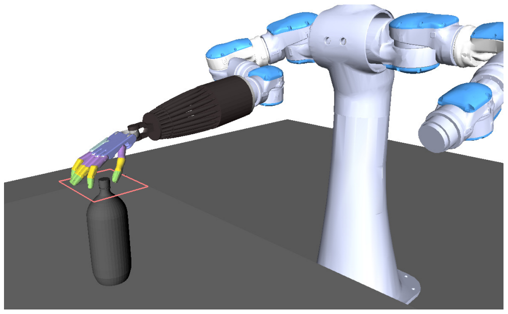

Figure 3.6: Adjusting the height of the control region to grasp non-flat objects

At first glance, the nature of our sticky-finger teleoperation system might seem to favor only flat objects or objects with flat surfaces, but this is generally not true. Overhead pick-and-place of non-flat objects for instance can be achieved simply by adjusting the height of the control region and hence the fingertips to an appropriate position over the target object for grasping. (Figure 3.6).

3.4 Hand and Arm Pose

A finger is said to be active whenever the finger is in contact with the multi-touch surface. For our multi-touch interface, the input at any given time is the set of active finger touches produced by the operator. This active set in turn specifies the corresponding active set of robot fingertips and their desired target positions in the workspace in the given time frame. Since the operator is required to register finger touches sequentially from the thumb to the little finger, assigning robot fingertips to the correct target positions is trivial. Our goal for each time-frame is then to compute a desirable robot hand pose that will allow these active robot fingertips to attain their respective targets. Our solution is to consider the robot's arm, palm and finger links as a whole kinematic skeleton rooted at the shoulder, with multiple branching finger chains. Given fingertip targets, we compute in one pass the inverse kinematics for the entire kinematic skeleton.

While an analytical inverse-kinematics solver has obvious benefits such as computational speed which is definitely valuable for real-time teleoperation, we opted for a numerical approach to solve our inverse kinematics problem. This is because, in a teleoperation task, it makes sense to have the robot fingertips track their desired targets as best as possible, even when they are out of reach. An analytical framework results in no movement if there is no inverse kinematics solution for a particular set of fingertip targets, while the incremental nature of a numerical-based method ensures that the end-effectors always move to reduce the error between the current fingertip positions and the desired targets, regardless of whether these targets are immediately reachable or not.

To solve our given problem, we first compute the active Jacobian of our system. The Jacobian J relates changes in our active joints $\dot{\mathbf{q}}$ to changes in our active fingertip positions $\dot{\mathbf{x}}$:

$$\dot{\mathbf{x}} = J\dot{\mathbf{q}} \tag{3.1}$$

In our case, $\dot{\mathbf{q}}$ has a dimension equal to the total number of degrees of freedoms in all the joints of the entire active kinematic skeleton, while $\dot{\mathbf{x}}$ has a dimension of 3 x a, where a is the current number of active fingers. Because the set of active sticky fingers can change at any one time, the dimensions of J are generally not constant as well.

Because we typically have a large number of joints in our active kinematic skeleton with respect to the degrees of freedom in our goal, our system is highly redundant. This results in the Jacobian being non-square and thus non-invertible. We can however use the Jacobian pseudoinverse method which gives the best approximate solution to (1) in terms of minimizing the residual norm $||J\dot{\mathbf{q}} - \dot{\mathbf{x}}||$ as well as $||\dot{\mathbf{q}}||$:

$$\dot{\mathbf{q}} = J^{\dagger}\dot{\mathbf{x}} \tag{3.2}$$

where $J^{\dagger}$ is the pseudoinverse of J. However, the pseudoinverse method returns only one possible solution, which might not always yield the most ideal arm and hand pose given the highly redundant nature of our system. In particular, we want a pose that is reasonably natural and free from collision. A more general solution to (1) allows us to constrain our solutions in terms of well-defined secondary tasks and is given the following equation:

$$\dot{\mathbf{q}} = J\dot{\mathbf{x}} + (I - J^{\dagger}J)\mathbf{z} \tag{3.3}$$

It can be shown that the operator $(I - J^{\dagger}J)$ projects an arbitrary vector $\mathbf{z}$ in joint-velocity space onto the nullspace of J, and by choosing z carefully, one can allow the system to attempt to satisfy desirable secondary constraints without affecting the primary task of attaining the original fingertip goal positions. Many previous authors have exploited this nullspace method to enforce certain secondary qualities in their solutions such as collision avoidance or joint limit avoidance [23][1].

We adopt the same approach for our system. For our case, we define a secondary goal that constrains the roll of the robot palm to zero. This constraint is essential because all the possible robot poses that our multi-touch interface supports always have the palm facing generally downwards, with the finger links extending downwards from the palm onto the workspace plane. If the constraint is not enforced, the palm can in some cases start to roll sidewards which may then cause the finger links to fail to reach their desired targets. We define this secondary goal as follows:

$$\dot{\mathbf{x_2}} = J_2\dot{\mathbf{q}} \tag{3.4}$$

where $\dot{\mathbf{x_2}}$ is the velocity of the palm roll and $\dot{\mathbf{q}}$ is the Jacobian defined at the palm origin with respect to all arm joints only. By substituting (4) into (2), then solving for $\mathbf{z}$, and finally plugging z back again into (2), we obtain the following equation:

$$\dot{\mathbf{q}} = J\dot{\mathbf{x}} + [J_2(I - J^{\dagger}J)]^{\dagger}\dot{\mathbf{x_2}} - J_2(J^{\dagger}\dot{\mathbf{x}})) \tag{3.5}$$

$\dot{\mathbf{q}}$ now yields the joint velocities that will bring us to the desired active fingertip goal positions while at the same time ensuring that the palm roll is as upright as possible.

After obtaining the new joint velocities for each time step, we check for possible collisions between all robot links as well as with the environment. We do not move any joints in a particular time step if any collision is detected. In the case where joint limits are exceeded, we simply clamp the joints at the appropriate limits. As a last note, the choice of the starting hand pose also affects the quality of subsequently computed poses, since the pseudoinverse approach minimizes change in joint angles from step to step. Therefore, we hand code a desirable starting pose, one that is considerably natural and free from nearby obstacles.

3.5 Implementation Details

3.5.1 Teleoperation System

We use the Apple iPad as our multi-touch device for teleoperation. We also rely on a TUIO-based open-source finger tracker application for iPad, TuioPad [3], which tracks finger positions on the touch-screen and sends normalized active finger coordinates as UDP packets to our host machine via wireless Ethernet. The host machine runs on an Intel Pentium 4 processor at 3.60 GHz. Our mapping program, which resides on this machine, listens on a suitable UDP port for possible touch events. During each time-frame, it converts the most currently received touch positions in touch screen coordinates to target fingertip world positions in the plane of the control region on the actual workspace, and computes the new inverse kinematics for the robot arm and hand joint positions. We use OpenRAVE [11] to model the simulated robot and the workspace environment. An open loop is run continuously at a frequency of 50Hz on the host which dispatches the latest computed joint positions to the arm and hand joint controllers during each iteration.

3.5.2 End-effector positions

In order to solve the inverse kinematics problem, we need to first assign a location on each fingertip link as the end-effector position. Essentially, this location will be the ideal point of *sticky-finger* contact for each fingertip. We assume that for each fingertip, this point does not change over the course of any manipulation task. A position that we found that works reasonably well is the middle point of the roundest part of each finger, with the exception of the thumb. For the latter, we assigned a location on the outer side of the thumb (Figure) because both the human and the robot thumb tend to be rolled outwards, away from the center of the hand by construct. We find these assignments reasonably consistent with the typical human fingertip regions that contact the multi-touch surface for many multi-touch gestures.

3.5.3 Lateral Scrolling

As described in Section III, the operator can translate the control region laterally by moving any fingertip against an edge of the multi-touch surface. We implemented this feature by checking for possible finger touches within a tiny margin on each side of the multi-touch screen in each time step. We also actively keep track of the last non-zero velocity for each active finger touch. The rationale for this is to able to scroll the control region diagonally beyond an edge when a finger approaches the particular edge at an angle; otherwise we are limited to strictly left, right, up and down moves for each scroll event. If a potential finger at a margin lingers for more than 0.5 seconds, we compute the angle from the last non-zero velocity seen for this finger and scroll the hand at a designated speed in this direction.

3.5.4 Vertical Scrolling

For vertical scrolling of the hand, we leverage the accelerometer that is built into the iPad to find the pitch of the device at a particular point of time. In particular, we are interested in the x-component of the acceleration vector, which is defined as the axis that runs vertically down the face of the device when the latter is in a landscape configuration. In order to eliminate noise, we first isolate the portion of the acceleration that is attributed to gravity from the portion that is caused by any device motion, by using a basic low-pass filter, as recommended by Apple [4]:

$$\mathbf{g_x} = \mathbf{x_{raw}} * \alpha + (\mathbf{g_{last}} * (1 - \alpha)) \tag{3.6}$$

where $\mathbf{g_x}$ is that x-component of the gravity vector we are interested in computing, $\mathbf{a_{raw}}$ is the x-component of the raw acceleration vector and $\mathbf{g_{last}}$ is the last filtered x-component of the gravity value. α is the smoothing factor, which specifies the relative proportion of unfiltered acceleration data and the previously filtered value to be used for smoothing. We used an α value of 0.1.

A positive $\mathbf{g_x}$ value implies that the side of the device nearer to the operator is tilted downwards while a negative $\mathbf{g_x}$ value means that this same side is being tilted upwards. A value close to 0 implies that the device is lying approximately flat. The control region is then moved at a fixed velocity v, depending on the interval that this value lies:

$$v = \begin{cases} v_s, & \text{if } \mathbf{g_x} > \theta. \\ -v_s, & \text{if } \mathbf{g_x} < -\theta. \\ 0, & \text{otherwise.} \end{cases} \tag{3.7}$$

where v_s is the scroll velocity and θ is a minimum threshold value for the onset of any movement.

3.6 Results and Evaluation

Figures 3.7 - 3.13 show various snapshots of teleoperated tasks achieved with simple finger gestures using our multi-touch interface, namely grasping, sliding, twisting and pick-and place tasks.

To evaluate our system, we carried out informal user tests. Our main goal was to uncover any possible difficulties and awkwardness that might be associated with using our teleoperation interface as well as to gauge the range of common household objects that the system can comfortably handle.

We designed two types of common telemanipulation tasks which we felt our interface would be very suited for, namely planar sliding and pick-and-place. For the sliding task, we used a variety of objects, including a floppy disk, a screwdriver, a pill bottle, a small spongy ball and a towel. The start position and goal position were randomized for each object and for each subsequent run. The direct distance between the start and goal position is 0.2m.

On the first round of sliding, most testers took more than 50 seconds on average to slide each given object; with a number failing to carry out some of the tasks under 1.5 minutes. However, almost all testers improved substantially with a second run, clocking a timing of less than 40 seconds on average across the 5 sliding tasks. Also, we observed that the floppy disk and the ball were more challenging to handle than the other objects due to their geometries; the disk felt too thin to be grasped or gripped in any way whilst for the ball, an incorrect approach of the hand could shift its rest position, resulting in additional grasp readjustment.

Interestingly, testers employed two human-inspired strategies to slide flat objects such as the floppy disk, namely pinning and gathering (Figure 3.8 and 3.9). Both were executed with a mixed amount of success amongst our testers. The first strategy, pinning, involves landing a number of fingers on the disk surface, in an attempt to pin the disk down before moving it. Gathering, on the other hand, is achieved by sliding a few fingers along the table surface until they strike an edge of the object, and then carefully pushing the object with these fingers to the goal position. Pinning was generally successful for testers who attempted to use two or more pinning fingers near the center of the disk to gain enough ”sticky” contact on the smooth disk surface. For gathering, successful execution seemed to depend on the spread of multiple fingers to make maximum contact across the edge of the object to be pushed. We observed that this strategy minimized object rotation, which is essential for the success of straight-line sliding tasks.

On the other hand, testers were very successful with the screwdriver, pill bottle and the towel, mainly because they could get a good grip on these objects. We observed that almost all testers used a pinch gesture involving at least 2 fingers to produce a corresponding pinch grasp on these objects.

Our next task required the testers to carry out pick-and-place tasks over a distance of 0.4m with a screw driver, a ball and a cup. The main goal was to test the intuitiveness of our lateral and vertical hand-scrolling features. While most testers were able to accomplish these scrolling tasks comfortably, some testers commented on the slight latency they felt between the gestures they performed and the corresponding robot finger motion, which caused them to overshoot the desired stop position when scrolling the hand. A few of them however attempted to compensate for the overshooting by stopping the scrolling slightly before the desired goal position.

In general, most testers reflected that the controls were easy to learn and natural because the mapping from their finger motions to the corresponding robot finger motions felt direct. Most testers were also confident of localizing the hand at a desired location fairly accurately, given a current position of the hand. However, a tester commented that the interaction could be more direct if the control region can be projected exactly onto the multi-touch screen itself. Another tester also suggested that the execution speeds could be improved. Lastly, a problem that we observed across some failure cases was the occurrence of accidental touches. These errant touches could sometimes result in a very awkward robot hand pose unless the user reacts quickly to re-register the correct fingers.

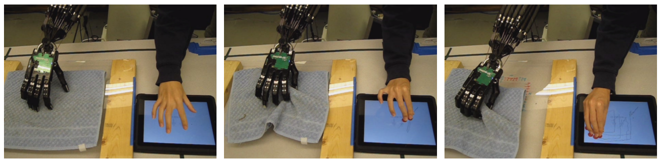

Figure 3.7: Pinching and dragging a towel using a simple pinch-and-slide gesture

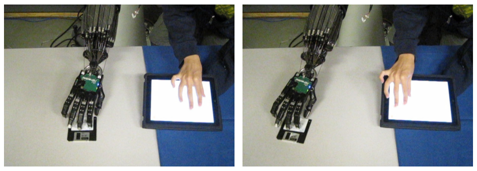

Figure 3.8: Sliding a light and flat floppy disk with pinning fingers

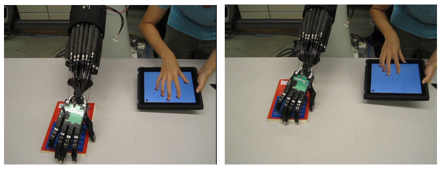

Figure 3.9: Sliding a book with ”gathering” fingers

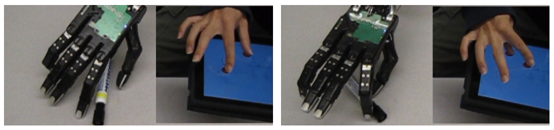
Figure 3.10: Two-finger rotation of a pen

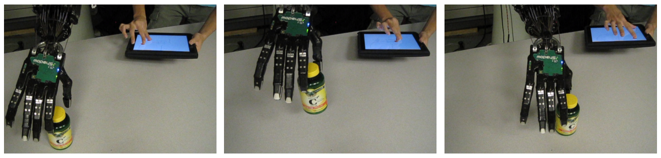
Figure 3.11: Pick-and-place task of a pill bottle using a two-fingered pinch

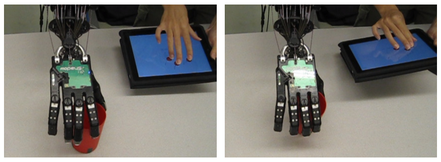
Figure 3.12: A power grasp on the opening of a cup using a four-fingered pinch gesture.

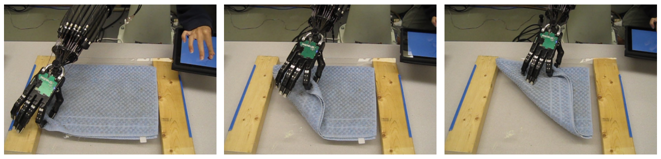
Figure 3.13: Folding a piece of cloth in a controlled fashion using: a pinch, move-up, translate, move-down and release sequence.

Chapter 4

Discussion and Future Work

4.1 Cloth Manipulation

We have presented a novel real-time system that allows the user to intuitively control and manipulate cloth with multiple fingers using a multi-touch device. Our techniques of sticky fingers and pinch-lifting, coupled with the use of Verlet Integration with iterative constraint satisfaction, enable the user to generate a range of interesting cloth interactions that are often not attainable with the conventional mouse device. We believe that our system will be very useful for interactive 3D scene design involving multiple cloths, virtual garment design as well as cloth folding applications. In addition, our interaction techniques can potentially be used to generate realistic animations of dynamic cloth manipulations, which can often be challenging to attain manually.

Our approach has several limitations. Firstly, since the nature of the multi-touch screen is two-dimensional, cloth can only be controlled along a single plane at any instance. Fingers cannot be lifted off the surface. While planar manipulations by sticky fingers can allow the user to generate rather interesting cloth behavior that would previously be challenging to generate, such as pinching, bunching, flattening, global cloth rotation and translation, many complex cloth manipulations like folding and draping count on the ability to lift the cloth off the underlying plane. We attempt to provide ”2.5-dimensional control” by introducing two modes of lifting: namely sticky-finger lifting and pinch-lifting. For sticky-finger lifting, we explictly provide a toggle key to switch the control plane from the default x-z plane to the y-z plane(or x-y plane) while for pinch-lifting, we switch these axes of control automatically. While we find these modes sufficient for a wide range of manipulations, a user may want to control cloth along a user-defined arbitrary plane other than the three preset ones. Perhaps, allowing manipulation of the control plane by interactively moving the camera might be more intuitive to the user.

Cloth-cloth collisions, both with itself, and with other cloths, are not implemented for performance reasons. While we think this decision is crucial to keep the system at real-time, perhaps an optional collision detection scheme such as basic particle-particle repulsion might appeal to certain users seeking for physical correctness over manipulability and real-time control. In fact, the proposed repulsion scheme may suffice for some interesting effects such as the ability to drape cloths over a laundry line modeled as a thin rope.

We are also considering the support of manipulations such as cutting and stitching, which we believe will be easy to incorporate into our current framework. With these operations, the user will be able to cut or sew cloth by simple finger swipe gestures along cloth surfaces. When combined with our supported operations such as folding and tearing, the user can acheive many interesting outcomes with the cloth.

Currently, our system permits only rectangular cloths. In future, we would also like to provide support for both gesture recognition and modeling of arbitrarily-shaped cloth. Instead of representing cloth as a regular grid of vertices, polygonal manifold meshes can be used to represent various cloth geometries more accurately.

In addition, our tearing model only supports cloth wire meshes and does not conserve total cloth area due to the direct removal of constraints, which visually represents the ”wire” between each particle during wireframe rendering. A more accurate model of tearing would allow us to render shaded cloth properly as well as conserve total area [27]. Also, if we can stick particles based on exact touch contact areas instead of assuming circular finger area contacts, the user will be able to tear cloth more realistically because the exact cloth area that is immobilized during tearing can be modeled.

We are also exploring the concept of ”non-sticky fingers”. Instead of sticking to sections of the cloth and moving them, such fingers slide along the surface of the cloth without adhering to any part of the cloth. In particular, ”non-sticky fingers” could be useful for smoothing out wrinkles on a cloth.

4.2 Telemanipulation

Our multi-touch interface for the teleoperation of a multi-fingered robot manipulator offers the operator a physically familiar sense of interaction, allowing him or her to directly telemanipulate objects in a natural and intuitive fashion using multiple fingertips. In addition, our system has the benefit of being portable, non-obstrusive and relatively inexpensive as compared to traditional bilateral teleoperation systems. We believe that our interface will be valuable for the teleoperation of common tabletop manipulations, such as pinching, sliding, twisting and pick-and-place tasks.

The teleoperation system has several limitations. Firstly, the interface only supports overhand poses - hand poses with the palm facing down. This limitation stems directly from the way one uses a multi-touch surface, which is essentially with the palm facing the surface. While we find overhand poses to be sufficient for a wide range of common tabletop manipulation tasks, especially planar ones, one might sometimes want to control the hand to grasp an object from the side instead of from the top, or slide the robot fingers along a sloped surface. A possible solution is to map the orientation of the device directly to the plane of the control region, but this means that a toggle or an alternative scheme to adjust the height of the control region is required since the current scheme already uses orientation information for height control.

The current inverse kinematics framework considers only fingertip positions as the goal in each time step because finger positions are the only information provided by the multi-touch device. A closer mapping to a robot hand pose can be computed if we have access to individual finger touch areas. We can then try to use these area information to deduce the orientation of a particular fingertip. For instance, a larger finger contact area might imply that the operator’s finger is close to being flushed with the surface while a small area might suggest an angle that is close to being perpendicular to the surface.

Currently, our system performs collision checking in every inverse kinematics step, and does not execute the computed joint velocities if they result in a potential collision. While non-execution in this case guarantees workspace safety, in rare cases however, the robot hand might seem like it has stalled, from the point of the operator. The stalling effect can be counter-intuitive especially if the operator believes that the robot hand should be able to mimic the gestures that were executed. While the operator might be able to learn to maneuver the hand out of its stuck state by trying some other finger motions, a better approach is to enforce additional control during the inverse kinematics move, such that the hand always steers clear from known obstacles. The most important obstacle to consider is often the workspace plane, which is usually well-defined. In future work, we consider adding collision avoidance as another secondary constraint to our inverse kinematics framework as described by Maciejewski [23]. This additional secondary constraint is responsible for pulling the hand in a direction perpendicular to the workspace plane if a point on a link comes close to the workspace. Since we have more than 2 constraints, a recursive formation of the nullspace method will have to be employed [5].

A useful extension to our system would be to have some form of force control for each active fingertip. This can be implemented if we are able to extract pressure information of each finger touch from the multi-touch surface and use it to control the pressure that is exerted by each robot fingertip on the target object surface. Interestingly, the operator also gains a limited but nevertheless useful form of sensing feedback from the reaction force exerted by the multi-touch surface on each fingertip with this extension.

Our framework can also be extended naturally to bimanual manipulators. Sticky fingers will work regardless of the number of finger touches, as long as we careful to map finger touches to the correct robot fingers. With two set of sticky-fingers, interesting planar telemanipulations such as tearing of paper and the folding of cloth by can be attained. We are also working on extending our sticky-finger framework to robot hands that is less human-like, such as the three-fingered Barrett Hand.

Bibliography

[1] Automatic supervisory control of the configuration and behavior of multibody mechanisms. *Systems, Man and Cybernetics, IEEE Transactions on*, 7(12):868 –871, dec. 1977. 3.1, 3.4

[2] M. Akten. MSA Remote for iPhone. `http://memo.tv/iphone/msaremote`. 2.4.1

[3] M. Akten and M. Kaltenbrunner. Tuiopad, open source tuio app for ios based on openframeworks. https://code.google.com/p/tuiopad/. 3.5.1

[4] Apple. Event handling guide for ios. http://developer.apple.com/library/iOS/. 3.5.4

[5] P. Baerlocher and R. Boulic. Task-priority formulations for the kinematic control of highly redundant articulated structures. In *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, volume 1, pages 323 –329 vol.1, oct 1998. 3.1, 4.2

[6] D. Baraff and A. Witkin. Large steps in cloth simulation. In *Computer Graphics (Proceedings of SIGGRAPH 98)*, 1998. 2.1

[7] D. E. Breen, D. H. House, and P. H. Getto. A physically based particle model of woven cloth. In *The Visual Computer*, 1992. 2.1

[8] R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact and friction for cloth animation. In *ACM Transactions on Graphics (Proc. ACM SIGGRAPH 2002)*, volume 21, pages 594–603, 2002. 2.1

[9] S. R. Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. Technical report, IEEE Journal of Robotics and Automation, 2004. 3.1

[10] K.-J. Choi and H.-S. Ko. Stable but responsive cloth. In *ACM Trans. Graph.*, volume 21, pages 604–611, 2002. 2.1

[11] R. Diankov. Openrave. http://openrave.programmingvision.com/. 3.5.1

[12] W. B. Griffin, R. P. Findley, M. L. Turner, and M. R. Cutkosky. *Calibration and Mapping of a Human Hand for Dexterous Telemanipulation*, pages 1145–1152. 2000. 3.1

[13] W. B. Griffin, W. R. Provancher, and M. R. Cutkosky. Feedback strategies for telemanipulation with shared control of object handling forces. *Presence: Teleoperators and Virtual Environments*, 14(6):720–731, 2005. 3.1

[14] M. Hancock, T. ten Cate, and S. Carpendale. Sticky tools: full 6dof force-based interaction for multi-touch tables. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '09, pages 133–140, New York, NY, USA, 2009. ACM. 3.1

[15] P. F. Hokayem and M. W. Spong. Bilateral teleoperation: An historical survey. *Automatica*, 42(12):2035 – 2057, 2006. 3.1

[16] H. Hu, J. Li, Z. Xie, B. Wang, H. Liu, and G. Hirzinger. A robot arm/hand teleoperation system with telepresence and shared control. In *Advanced Intelligent Mechatronics. Proceedings, 2005 IEEE/ASME International Conference*, pages 1312–1317, 2005. 3.1

[17] T. Igarashi and J. F. Hughes. Clothing manipulation. In *UIST and I3D Reprise session, ACM SIGGRAPH 2003 San Diego, USA*, page 697, July 2003. 2.1

[18] F. D. IX, H. Qin, and A. Kaufman. A novel haptics-based interface and sculpting system for physics-based geometric design. *INTERNATIONAL JOURNAL OF SHAPE MODELING*, 2, 2001. 2.1

[19] T. Jakobsen. Advanced character physics. In *Game Developer's Conference*, 2001. 2.1, 2.3

[20] M. Kaltenbrunner, T. Bovermann, R. Bencina, and E. Costanza. TUIO - a protocol for table based tangible user interfaces. In *Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation (GW 2005), Vannes, France*, 2005. 2.4.1

[21] J. Kato, D. Sakamoto, M. Inami, and T. Igarashi. Multi-touch interface for controlling multiple mobile robots. In *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, CHI EA '09, pages 3443–3448, New York, NY, USA, 2009. ACM. 3.1

[22] J. Kofman, V. Siddharth, W. Xianghai, and L. Timothy. Teleoperation of a robot manipulator from 3d human hand-arm motion kofman. In *Proceedings of SPIE - The International Society for Optical Engineering*, volume 5264, pages 257–265, 2003. 3.1

[23] A. A. Maciejewski and C. A. Klein. Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. *The International Journal of Robotics Research*, 4(3):109–117, 1985. 3.1, 3.4, 4.2

[24] Y. Meng, P. Y. Mok, and X. Jin. Interactive virtual try-on clothing design systems. In *Computer-Aided Design*, volume 42, April 2010. 2.1

[25] M. Meyer, G. Debunne, M. Desbrun, and A. H. Barr. Interactive animation of cloth-like objects in virtual reality. *The Journal of Visualization and Computer Animation*, 12(1):1–12, May 2001. 2.1

[26] M. Micire, M. Desai, J. L. Drury, E. McCann, A. Norton, K. M. Tsui, and H. A. Yanco. Design and validation of two-handed multi-touch tabletop controllers for robot teleoperation. In *Proceedings of the 16th international conference on Intelligent user interfaces*, IUI '11, pages 145–154, New York, NY, USA, 2011. ACM. 3.1

[27] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff". Position based dynamics. In *Proceedings of Virtual Reality Interactions and Physical Simulations (VRIPhys)*, pages 71–80, November 2006. 2.1, 2.3.4, 2.3.5, 4.1

[28] NUIGroup. Touchlib, A Multi-Touch Development Kit. `http://nuigroup.com/touchlib/`. 2.4.1

[29] X. Provot. Deformation constraints in a mass-spring model to describe rigid cloth behaviour. In *Proceedings of Graphics Interface 1995*, 1995. 2.1, 2.3.1

[30] J. L. Reisman, P. L. Davidson, and J. Y. Han. A screen-space formulation for 2d and 3d direct manipulation. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, UIST '09, pages 69–78, New York, NY, USA, 2009. ACM. 3.1

[31] R. Rohling and J. Hollerbach. Optimized fingertip mapping for teleoperation of dextrous robot hands. In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pages 769 –775 vol.3, may 1993. 3.1

[32] S. Sandler. MTmini-DIY Multitouch Mini Pad. `http://sethsandler.com/multitouch/mtmini/`. 2.4.1

[33] T. B. Sheridan. *Telerobotics, Automation, and Human Supervisory Con- trol.* The MIT Press, Cambridge, MA, 1992. 3.1

[34] M. Stilman, K. Nishiwaki, and S. Kagami. Humanoid teleoperation for whole body manipulation. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 3175 –3180, may 2008. 3.1

[35] D. Terzopoulos, J. J. Platt, A. Barr, and K.Fleischer. Elastically deformable models. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, 1987. 2.1

[36] M. L. Turner, R. P. Findley, W. B. Griffin, M. R. Cutkosky, and D. H. Gomez. Development and testing of a telemanipulation system with arm and hand motion. In *Proc. ASME Dynamic Systems and Control Division (Symposium on Haptic Interfaces for Virtual Environments and Teleoperators)*, volume DSC-Vol. 69-2, pages 1057–1063, 2000. 3.1

[37] A. D. Wilson, S. Izadi, O. Hilliges, A. Garcia-Mendoza, and D. Kirk. Bringing physics to the surface. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, UIST '08, pages 67–76, New York, NY, USA, 2008. ACM. 3.1

[38] C. Wojtan, P. J. Mucha, and G. Turk. Keyframe control of complex particle systems using the adjoint method. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 15–23, 2006. 2.1